IN THE CLAIMS

Please make the following amendments to the claims:

1. (Currently Amended) A processor for offloading processing in a storage environment, comprising:

a ~~processor~~ zero bus turnaround ("ZBT") interface that interfaces said processor to a network processor configured to perform a storage function; and

semaphore circuitry, coupled to said ~~processor~~ ZBT interface, that receives a signal from said network processor, and that controls a semaphore related to said signal for locking and unlocking access to data.

2. (Original) The processor of claim 1, wherein said semaphore circuitry manages a queue for access to said semaphore.

3. (Original) The processor of claim 2, wherein said semaphore circuitry receives a second signal from said network processor and removes a request from said queue in response to said second signal when said network processor no longer desires said semaphore.

4. (Original) The processor of claim 2, wherein said semaphore circuitry refrains from sending to said network processor a second signal indicating said semaphore is unavailable, whereby said network processor continues to wait for said semaphore and said semaphore circuitry maintains ordered access to said queue.

5. (Original) The processor of claim 1, wherein said signal comprises one of a plurality of access requests for one of a plurality of semaphores, wherein said

semaphore circuitry manages said plurality of access requests in a plurality of queues, and wherein each of said plurality of queues corresponds to a respective one of said plurality of semaphores.

6. (Original) The processor of claim 1, wherein said semaphore circuitry comprises:

a command queue that stores said signal received from said network processor.

7. (Original) The processor of claim 1, wherein said semaphore is a structure in a hash array, and wherein said semaphore circuitry comprises:

a hash key generator that performs a hashing function on said signal for accessing said hash array.

8. (Original) The processor of claim 1, wherein said semaphore circuitry comprises:

an update engine that receives a second signal from said network processor relating to a first process thread on said network processor, releases a lock on said semaphore related to said second signal, and sends a third signal to said network processor associating said semaphore with a second process thread on said network processor.

9. (Original) The processor of claim 1, wherein said semaphore circuitry comprises:

a semaphore queue manager that manages a queue of a plurality of semaphores.

10. (Original) The processor of claim 1, wherein said semaphore circuitry manages a queue for access to said semaphore, wherein said semaphore circuitry comprises:

a hash key generator that performs a hashing function on said signal for accessing a hash array, and that generates a hash key related to said signal, wherein said semaphore is a structure in said hash array, and wherein said signal relates to a first process thread on said network processor; and

an update engine, coupled to said hash key generator, that receives said hash key, that releases a lock on said semaphore related to said hash key, and that sends a second signal to said network processor associating said semaphore with a second process thread on said network processor, wherein said first process thread and said second process thread are associated with said semaphore in said queue.

11. (Currently Amended) A method of controlling a processor for offloading processing in a storage environment, comprising the steps of:

receiving a signal from a network processor configured to perform a storage function through a zero bus turnaround ("ZBT") interface;

controlling a semaphore related to said signal for locking and unlocking access to data.

12. (Original) The method of claim 11, wherein said step of controlling said semaphore includes:

managing a queue for access to said semaphore.

13. (Original) The method of claim 12, further comprising:

receiving a second signal from said network processor; and

removing a request from said queue in response to said second signal when said network processor no longer desires said semaphore.

14. (Original) The method of claim 12, further comprising:

refraining from sending to said network processor a second signal indicating said semaphore is unavailable, whereby said network processor continues to wait for said semaphore and said processor maintains ordered access to said queue.

15. (Original) The method of claim 11, wherein said signal comprises one of a plurality of access requests for one of a plurality of semaphores, and wherein said step of controlling said semaphore includes:

managing said plurality of access requests in a plurality of queues, wherein each of said plurality of queues corresponds to a respective one of said plurality of semaphores.

16. (Original) The method of claim 11, wherein said step of controlling said semaphore includes:

receiving a second signal from said network processor relating to a first process thread on said network processor;

releasing a lock in said semaphore related to said second signal; and

sending a third signal to said network processor associating said semaphore with a second process thread on said network processor.

17. (Original) The method of claim 11, wherein said signal comprises one of a plurality of access requests for one of a plurality of semaphores, and wherein said step of controlling said semaphore includes:

managing said plurality of access requests in a plurality of queues, wherein each of said plurality of queues corresponds to a respective one of said plurality of semaphores;

receiving a second signal from said network processor relating to a first process thread on said network processor;

releasing a lock on said semaphore related to said second signal;

and sending a third signal to said network processor associating said semaphore with a second process thread for said network processor.

18-51. (Canceled)

52. (New) A storage server comprising:

a network processor to control routing of data frames into and out of the storage server;

a co-processor to offload some functions performed by the network processor, the co-processor to communicate with the network processor via at least one interface; and

a memory to store data for the co-processor, the memory coupled to the co-processor, wherein

the co-processor is to receive a command to release a semaphore from the network processor and return a release acknowledgement response if a thread is waiting for the semaphore.

53. (New) The storage server of claim 52 wherein the co-processor is a field-programmable gate array ("FPGA") or a programmable logic device ("PLD").

54. (New) The storage server of claim 52, further comprising:

a host processor to configure and monitor the co-processor, the host processor connected to the co-processor via a separate interface.

55. (New) A method comprising:

receiving a semaphore operation command from a network processor through an intra-system interface, the command to identify one of a plurality of semaphores;

updating a data structure in a memory according to the semaphore operation command; and

returning a semaphore operation result to the network processor through an intra-system interface, the result to indicate an outcome of the command.

56. (New) The method of claim 55, further comprising:

delaying the returning operation until after a second semaphore operation command from the network processor has been completed.

57. (New) The method of claim 55 wherein the intra-system interface is a zero bus turnaround ("ZBT") interface.